Unidad 2: Paso 2 – Identificar las estructuras básicas de programación

Cesar Alfonso Pallares Trespalacios

Cod: 88168768

Daira Margely Bermeo Ceidel

1117513761

Daniel Antonio Ruíz Carreño

Cod: 80228949

Heli Ceballos Amaya

Cod: 12278770

Luis Jonalber Fernández

Cod: 1094366

Presentad al tutor:

Jaime Rubiano Llorente

Grupo: 201416_47

Universidad Nacional Abierta y a Distancia

Universidad Ciencias Básicas, Tecnología e Ingeniería

2017

Contenido

INTRODUCCIÓN

En este trabajo presentamos las estructuras condicionales que hacen referencia a la toma lógica de decisiones para realizar alguna tarea en caso de cumplirse una o varias de las alternativas u opciones posibles. Este tipo de situaciones las aplicamos a diario y son muy comunes, puesto que por naturaleza es muy complicado realizar varias acciones de forma simultánea. En el campo de la programación es la situación similar, puesto que la aplicación de este criterio garantiza en correcto funcionamiento de una aplicación.

Las estructuras condicionales se clasifican de acuerdo al número de alternativas posibles, estas son: Simples. Compuestas. Múltiples. Anidadas

1. ESTRUCTURAS DE CONDICIONALES IF ELSE SIMPLES Y ANIDADAS

Las estructuras condicionales o selectivas se utilizan para tomar decisiones lógicas, las preguntas se platean por medios de condiciones estructuradas, y resultado será verdadero o falso, pero no ambas a la vez, en función del resultado será el bloque de acciones a ejecutar. Las instrucciones selectivas se clasifican en simples, dobles y múltiples.

Las palabras claves para las estructuras condicionales en Visual Basic, son: If, Then, Else, End, Case.

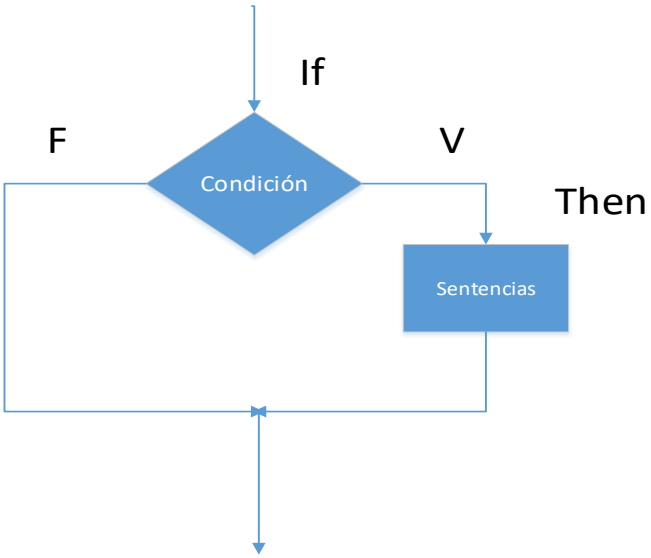
Los procedimientos de Visual Basic pueden probar condiciones y luego, dependiendo de los resultados de esa prueba, realizar operaciones diferentes. Las estructuras de decisión que soporta Visual Basic incluyen:

- If...Then
- If...Then...Else
- Select Case

Instrucción Selectiva Simple If Then

Se dice que la instrucción selectiva es simple si al cumplirse la condición se ejecuta una instrucción o bloque de instrucciones y en caso de ser falsa no se ejecuta alguna acción.

Representación gráfica



Sintaxis

If condition Then statement

End If

Elementos

- ✓ **Condición**

Requerido. Expresión. Debe evaluarse en True o False, o en un tipo de datos que sea implícitamente convertible a Boolean.

- ✓ **Then**

Obligatorio en la sintaxis de una línea, opcional en la sintaxis de varias líneas.

- ✓ **Statements**

Se ejecutan una o más instrucciones que siguen a If...Then que se ejecutan si condition se evalúa como True.

- ✓ **End If**

Termina el bloque If...Then...Else.

Ejemplo:

Supongamos que el grado de aprobación en un examen es 60 (de 100). A continuación, el código de Visual Basic.

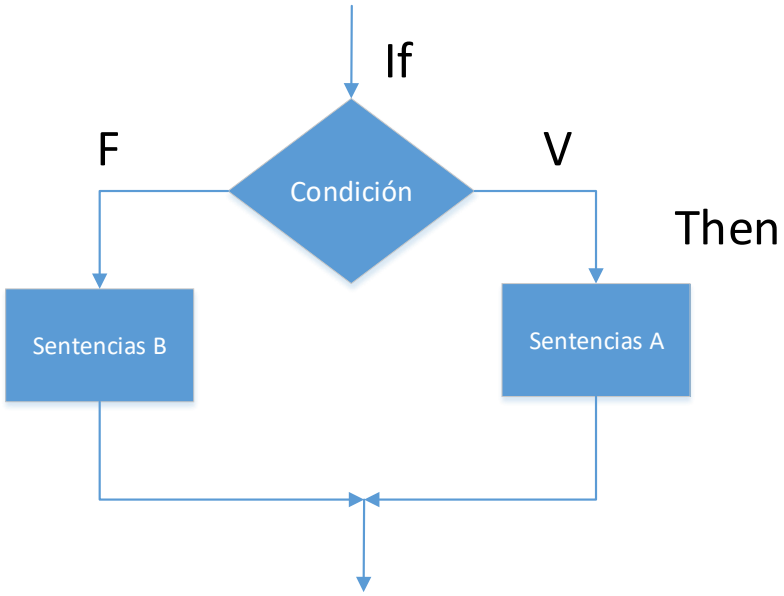
```
If studentGrade >= 60 Then
   Console.WriteLine("Aprobado")
End If
```

Determina si la condición studentGrade> = 60 es verdadera o falsa. Si la condición es verdadera, entonces se imprime " Aprobado " y se realiza la siguiente instrucción en orden. Si la condición es falsa, se omite la instrucción Console.WriteLine y se realiza la siguiente instrucción en orden. Se puede tomar una decisión en cualquier expresión que evalúe un valor del tipo Booleano de Visual Basic (es decir, cualquier expresión que se evalúe como True o False).

Instrucción Selectiva Doble If Then Else

Se utiliza cuando se tienen dos opciones de acción. Con la bifurcación doble se ejecuta un bloque de instrucciones A si se cumple la condición o bien se ejecuta el bloque de instrucciones B en caso de que no se cumpla, debido a que son naturalmente excluyentes.

Representación gráfica



Sintaxis

```
If condition1 Then
[statementblock-1]
Else
[statementblock-n]]
End If
```

Elementos

- ✓ **Condición**

Requerido. Expresión. Debe evaluarse en True o False, o en un tipo de datos que sea implícitamente convertible a Boolean.

- ✓ **Then**

Obligatorio en la sintaxis de una línea, opcional en la sintaxis de varias líneas.

- ✓ **Statements**

Se ejecutan una o más instrucciones que siguen a If...Then que se ejecutan si condition se evalúa como True.

- ✓ **Elseifcondition**

Expresión. Debe evaluarse en True o False, o en un tipo de datos que sea implícitamente convertible a Boolean.

- ✓ **Elseifstatements**

Se ejecutan una o más instrucciones que siguen a ElseIf...Then que se ejecutan si elseifcondition se evalúa como True.

- ✓ **Elsestatements**

Una o más instrucciones que se ejecutan si ninguna expresión condition o elseifcondition anterior se evalúa como True.

- ✓ **End If**

Termina el bloque If...Then...Else.

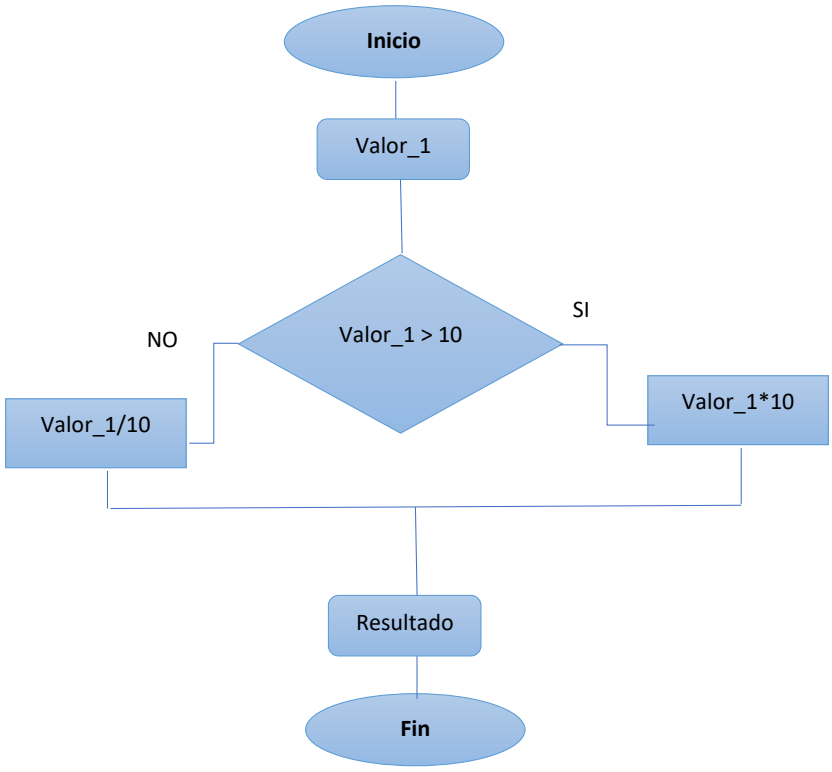
Ejemplo:

Un programa que calcula el valor si el numero es mayor o diferente de 10 lo multiplica en 2 y si es menor o igual a 10 lo divide, arrojando el resultado:

Código:

```
Private Sub calcular_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles calcular.Click
    If valor_1.Text > 10 Then
       resultado.Text = Val(valor_1.Text * 2)
    Else
       resultado.Text = Val(valor_1.Text / 2)
    End If
 End Sub
```

Diagrama de flujo

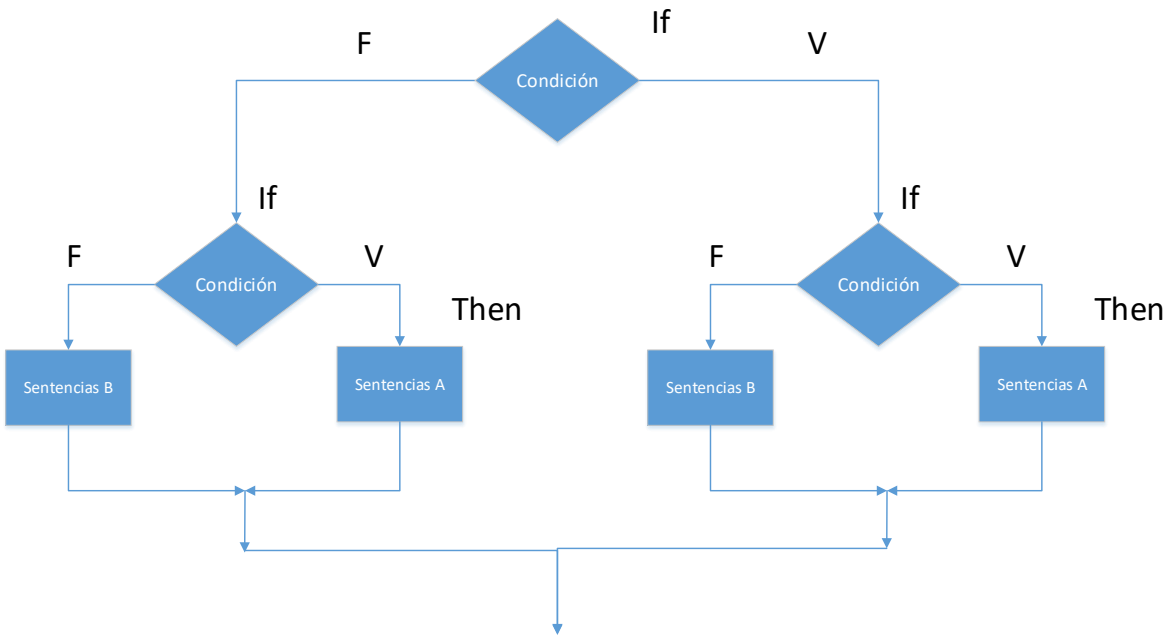


Instrucciones Selectivas Anidadas If Then Else

Existen programas donde se debe utilizar una sentencia selectiva, dentro de otra y puede que esta, este dentro de otra… En estos casos se dice que el programa tiene Ifs anidados.

Visual Basic primero prueba condición1. Si es False, Visual Basic procede a probar condition2, y así sucesivamente, hasta que encuentre una condición True. Cuando encuentra una condición True, Visual Basic ejecuta el bloque de instrucciones correspondiente y, a continuación, ejecuta el código después de End If. Como opción, puede incluir un bloque de sentencia Else, que Visual Basic ejecuta si ninguna de las condiciones es True.

Representación gráfica



Sintaxis

If condition1 Then
[statementblock-1]
[ElseIf condition2 Then

[statementblock-2]] ...

[Else

[statementblock-n]]

End If

Elementos

- ✓ **Condición**

Requerido. Expresión. Debe evaluarse en True o False, o en un tipo de datos que sea implícitamente convertible a Boolean.

- ✓ **Then**

Obligatorio en la sintaxis de una línea, opcional en la sintaxis de varias líneas.

- ✓ **Statements**

Se ejecutan una o más instrucciones que siguen a If...Then que se ejecutan si condition se evalúa como True.

- ✓ **Elseifcondition**

Expresión. Debe evaluarse en True o False, o en un tipo de datos que sea implícitamente convertible a Boolean.

- ✓ **Elseifstatements**

Se ejecutan una o más instrucciones que siguen a ElseIf...Then que se ejecutan si elseifcondition se evalúa como True.

- ✓ **Elsestatements**

Una o más instrucciones que se ejecutan si ninguna expresión condition o elseifcondition anterior se evalúa como True.

- ✓ **End If**

Termina el bloque If...Then...Else.

Ejemplo:

Determine el día actual de la semana y la hora del día, Vuelva verdadero si miércoles de 2 a 4 P.M., o si jueves de mediodía a 1 P.M.

```
Private Function CheckIfTime() As Boolean
    Dim dayW As DayOfWeek = DateTime.Now.DayOfWeek
    Dim hour As Integer = DateTime.Now.Hour

    If dayW = DayOfWeek.Wednesday Then
        If hour = 14 Or hour = 15 Then
            Return True
        Else
            Return False
        End If
    ElseIf dayW = DayOfWeek.Thursday Then
        If hour = 12 Then
            Return True
        Else
            Return False
        End If
    Else
        Return False
    End If
End Function
```

Las cláusulas ElseIf y Else son opcionales. Puede tener tantas cláusulas ElseIf como desee en una instrucción If...Then...Else, pero no puede aparecer ninguna cláusula ElseIf después de una cláusula Else. If ...Then...Else las instrucciones se pueden anidar una dentro de otra.

Sintaxis de una única línea

Puede utilizar la sintaxis de una sola línea para pruebas cortas y sencillas. Sin embargo, la sintaxis de varias líneas proporciona más estructura y flexibilidad y, generalmente, es más fácil de leer, mantener y depurar.

Lo que sigue a la palabra clave Then se examina para determinar si una declaración es un If de una sola línea. Si aparece cualquier otra cosa que no sea un comentario después de Then en la misma línea, ésta se trata como una instrucción If de una sola línea. Si no está presente Then, debe ser el comienzo de una instrucción If...Then...Else de varias líneas.

En la sintaxis de una línea, puede que se ejecuten varias instrucciones como resultado de una decisión If...Then. Todas las instrucciones deben estar en la misma línea y separarse con dos puntos.

Ejemplo:

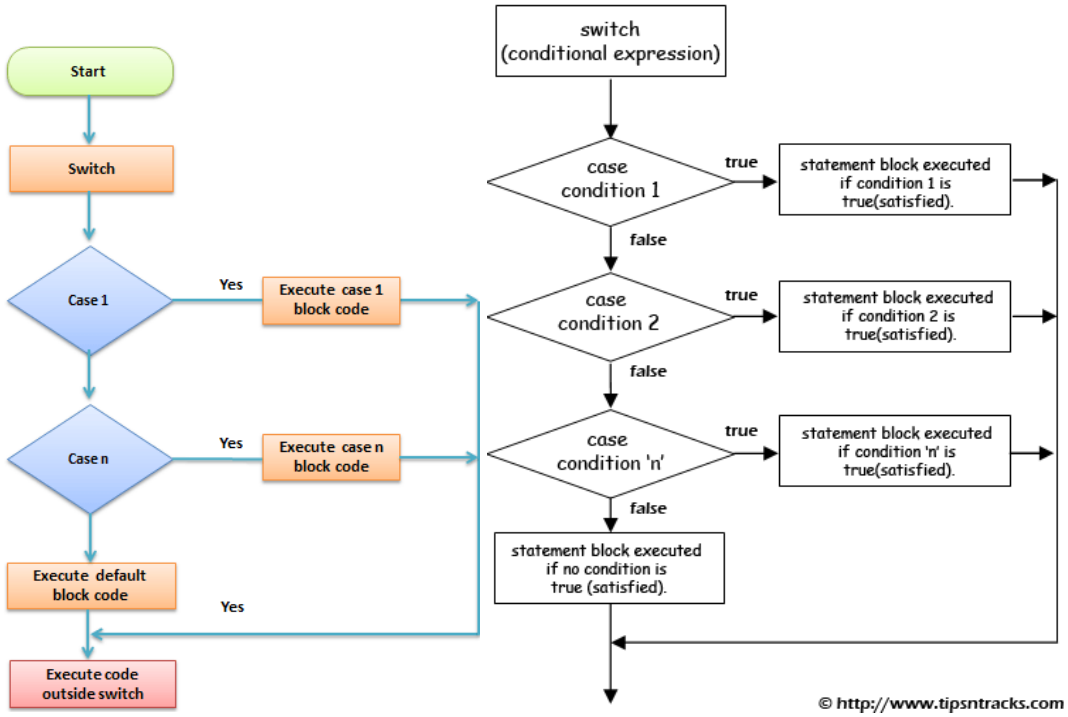
Si A> 10, ejecuta las tres declaraciones separadas por dos puntos en el orden que aparecen.

```
If A > 10 Then A = A + 1 : B = B + A : C = C + B
```

2. ESTRUCTURA CONDICIONAL CASE

Es una estructura de control donde evalúa una expresión, ejecuta uno de varios grupos de instrucciones, según el valor que cumpla la conducción. (DesarrolloWeb, 2017)

Diagrama



Sintaxis

Select Case Expresion1

Case Condicion1

BloquedeProcesos

Case Condicion2

BloquedeProcesos

Case Else

BloquedeProcesos

End Select

Se inicia con las palabras exclusivas Select Case, y se coloca la expresión a evaluar, Después de coloca los bloques de condiciones a los cuales se van a evaluar, también existe una sentencia cuando la expresión no coincide con ninguna. Las condiciones se evalúan de izquierda a derecha, y al final se cierra el bloque.

Ejemplo:

Inicio

Leer Edad

Select Case Edad

Caso 0 **a** 3

Mostrar “Es un bebe”

Case 4 **a** 11

Mostrar “Eres un niño”

Case 12 **a** 17

Mostrar “Eres un adolecente”

Case 18 **a** 200

Mostrar “Ya eres un adulto”

Case Else

Mostrar “Valor invalido”

Fin Select

Fin

En el algoritmo anterior se observa que se evalúa la edad, con la sentencia Select Case, donde si cumple algún valor, este ejecuta un bloque de comando.

Ahora veremos cómo se interpreta la sentencia Case en un algoritmo

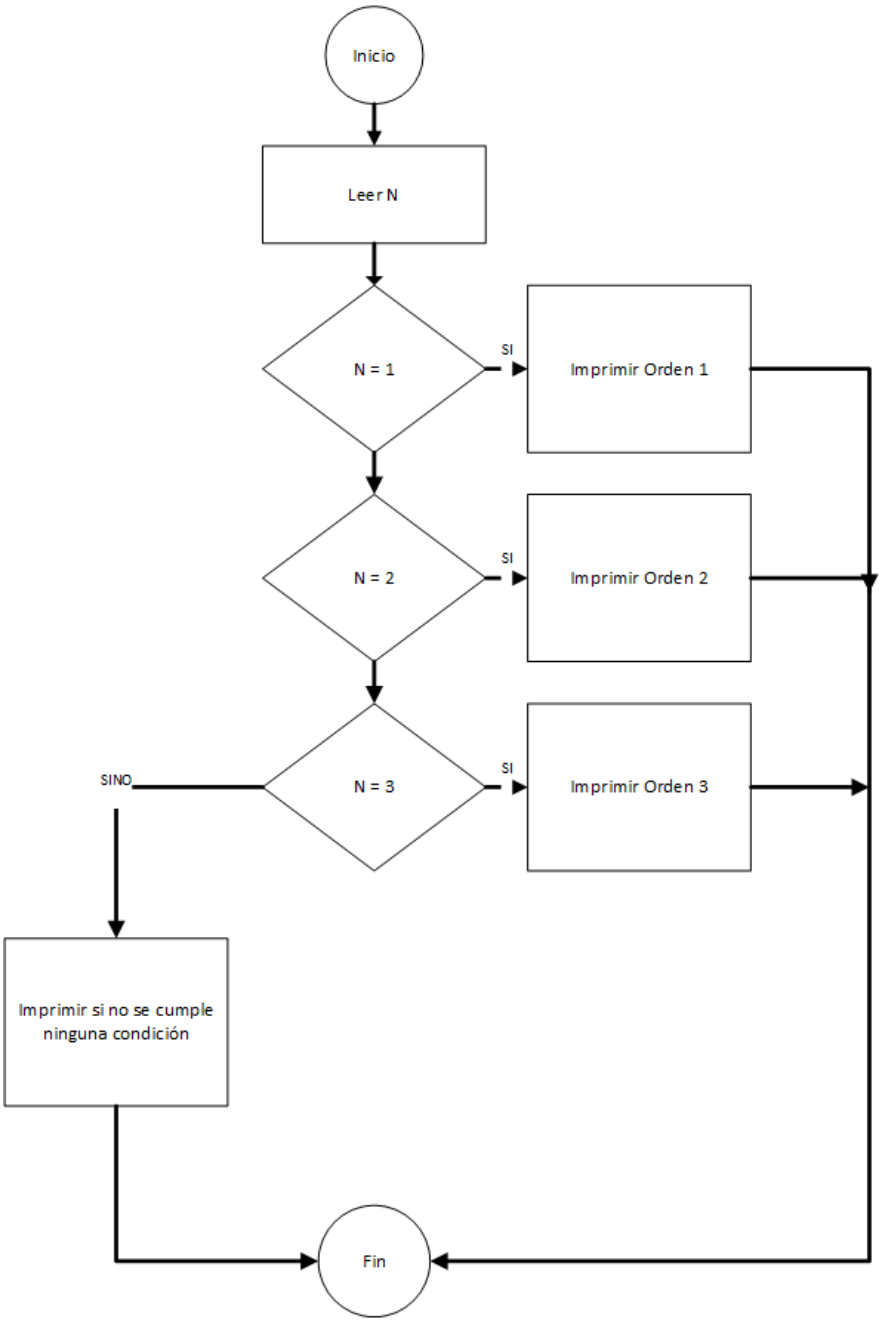


Figura 1: Alegorismo – sentencia Case

Se observa, que se evalúa la condición, si se cumple alguna de estas, se procede a ejercitar el bloque de secuencias para esta condición, si no se cumple y hay un SINO, se ejecuta este, si no existe esta condición, se da por terminado la sentencia.

La instrucción Select Case permite utilizar tantas condiciones (o casos) como sea necesario, y conviene escribir el código para situaciones en las que hay muchas opciones. Por ejemplo, suponga que el programa utilizó una variable String para almacenar una opción de color y se necesitaba obtener el valor de color. El código para la instrucción Select Case podría ser similar al siguiente:

```
Select Case Color
  Case "red"
     MsgBox("You selected red")
  Case "blue"
     MsgBox("You selected blue")
  Case "green"
     MsgBox("You selected green")
End Select
```

Case Else

La instrucción Case Else se puede utilizar para ejecutar el código cuando no se encuentra ninguna coincidencia, como en el siguiente ejemplo.

```
Select Case Color
  Case "red"
     MsgBox("You selected red")
  Case "blue"
     MsgBox("You selected blue")
  Case "green"
     MsgBox("You selected green")
  Case Else
     MsgBox("Please choose red, blue, or green")
End Select
```

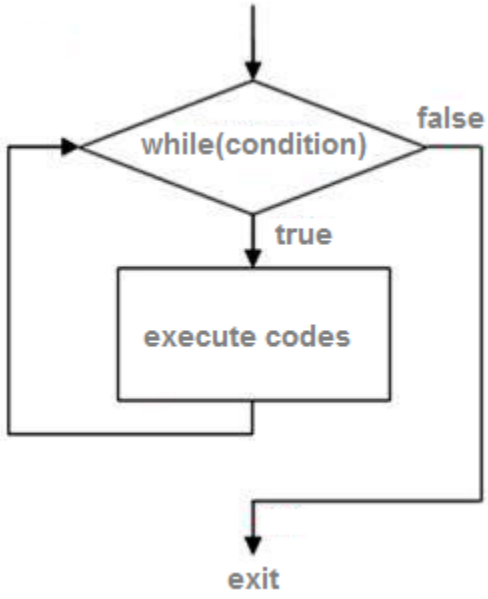
3. ESTRUCTURA DE CONTROL WHILE

La construcción While End While ejecuta un conjunto de instrucciones mientras la condición especificada en la instrucción While sea True.
Ejecuta una serie de instrucciones siempre que una condición dada sea verdadera

Partes del while

Término	Definición
condition	**Obligatorio. Expresión Boolean. Si condition es Nothing, Visual Basic la trata como False.**
statements	**Opcional. Una o más instrucciones a continuación de While, que se ejecutan cada vez que condition es True.**
Exit While	**Opcional. Transfiere el control fuera del bloque While.**
End While	**Obligatorio. Termina la definición del bloque While.**

Diagrama de flujo



Sintaxis

```
While condition
   [ statements ]
   [ Exit While ]
   [ statements ]
```

End While

Reglas del while

- Naturaleza de la condición. Generalmente, la condición es el resultado de comparar dos valores, pero también puede ser cualquier expresión que da como resultado un valor Boolean (Tipo de datos, Visual Basic) (True o False). Esto incluye los valores de otros tipos de datos, como los numéricos, que han sido convertidos a valores de tipo Boolean.

- Probar la condición. La instrucción While comprueba siempre la condición antes de iniciar el bucle. La ejecución en bucle continúa mientras el resultado de la condición sea True.

- Número de iteraciones. Si condition es False cuando se entra en el bucle por primera vez, ni siquiera se ejecuta una vez.

- Bucles anidados. Se pueden anidar bucles While colocando un bucle dentro de otro. También puede anidar distintos tipos de estructuras de control dentro de otras. Para obtener más información, vea Estructuras de control anidadas (Visual Basic).

- Transferir fuera del bucle. Exit (Instrucción, Visual Basic) transfiere el control inmediatamente a la instrucción que sigue a la instrucción End While. Por ejemplo, puede ser conveniente salir de un bucle si se detecta una condición que hace que sea innecesario o imposible continuar la iteración, como puede ser un valor erróneo o una solicitud de finalización. Puede poner cualquier número de instrucciones Exit While en cualquier lugar del bucle While. Exit While se utiliza a menudo después de evaluar alguna condición, por ejemplo, en una estructura If...Then...Else.

- Bucles sin fin. Un uso de Exit While consiste en comprobar una condición que podría ocasionar un bucle sin fin; es decir, un bucle que pudiera ejecutarse un número extremadamente elevado, o incluso infinito, de veces. Si detecta este tipo de condición, puede utilizar Exit While para salir del bucle. Para obtener más información,

Ejemplo 1

Realiza un ciclo while antes de cargar el form1

```
Public Class Form1
    Dim cont As Integer
```

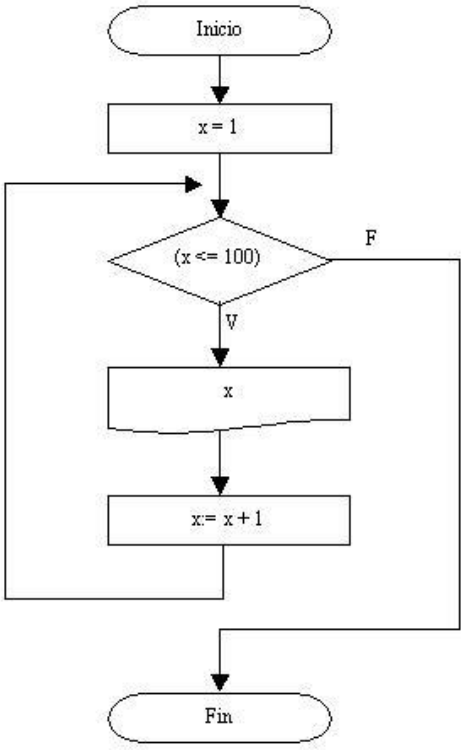
```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim counter As Integer = 0
        While counter < 20
            counter += 1

        End While
        MsgBox("While " & CStr(counter) & " times")

    End Sub
‘ realiza un ciclo while al presionar el botón e imprime el cont en el listbox1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        cont = 0
        While cont <= 10
            ListBox1.Items.Add(cont)
            cont = cont + 1
        End While

    End Sub
End Class
```

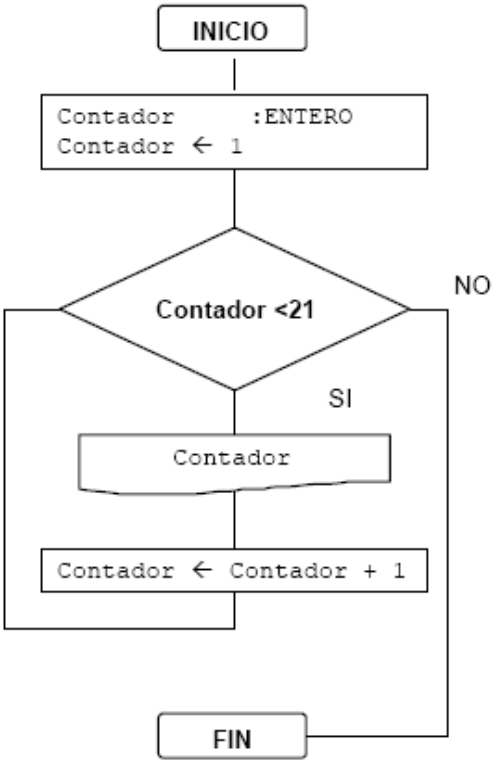
Ejemplo 2

```
Public Class Form1
    Dim num As Integer
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim I, num, resultado As Integer
        Dim Mensaje As Object
        I = 1
        While (I <= 3)
            Mensaje = InputBox("Que numero quiere elevar a la 3")
            num = Val(Mensaje)
            resultado = num ^ 3
            MsgBox("su resultado es" & resultado)
            I = I + 1
        End While
    End Sub
```

```
    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs)

    End Sub
End Class
```

Ejemplo con diagrama





4. ESTRUCTURA DE CONTROL DO WHILE

La estructura do while es otra estructura repetitiva, la cual ejecuta al menos una vez su bloque repetitivo, a diferencia del while o del for que podían no ejecutar el bloque.
Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo.

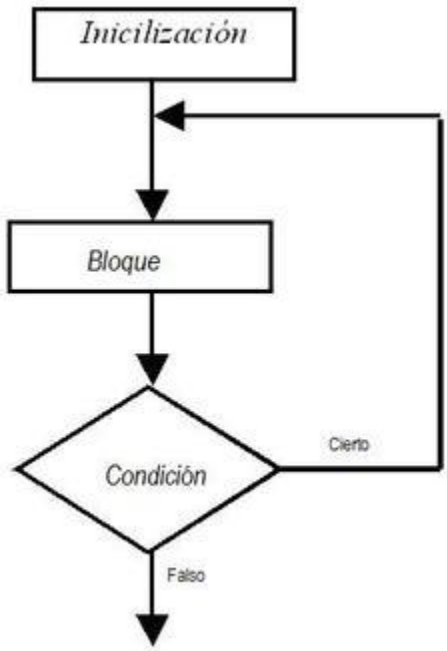
Al igual que en el bloque while, la condición de salida ha de ser una sentencia que devuelva un valor booleano, y esta puede ser el valor booleano sí, verdadero (true) si la condición se cumple, o falso si esta no se cumple (false). También puede contener el nombre de una variable booleana, y el valor de la expresión dependerá de su contenido. Se debe tener en cuenta que además de las variables también puede haber llamadas a funciones que devuelvan un valor.

PARTES DEL DO WHILE

Término	Definición
Do	**Requerido. Inicia la definición del bucle Do.**
While	**Obligatorio a menos que se utilice Until. Repite el bucle hasta que condition sea False.**
Until	**Obligatorio a menos que se utilice While. Repite el bucle hasta que condition sea True.**
condition	**Opcional. Expresión Boolean. Si condition es Nothing, Visual Basic la trata como False.**
statements	**Opcional. Una o más instrucciones que se repiten mientras o hasta que condition sea True.**
Continue Do	**Opcional. Transfiere el control a la siguiente iteración del bucle de Do .**
Exit Do	**Opcional. Transfiere el control fuera del bucle Do.**

Loop	**Requerido. Termina la definición del bucle Do.**

Diagrama de flujo



SINTAXIS

```
Do { While | Until } condition
   [ statements ]
   [ Continue Do ]
   [ statements ]
   [ Exit Do ]
   [ statements ]
```

```
Loop
-or-
Do
   [ statements ]
   [ Continue Do ]
   [ statements ]
   [ Exit Do ]
   [ statements ]
Loop { While | Until } condition
```

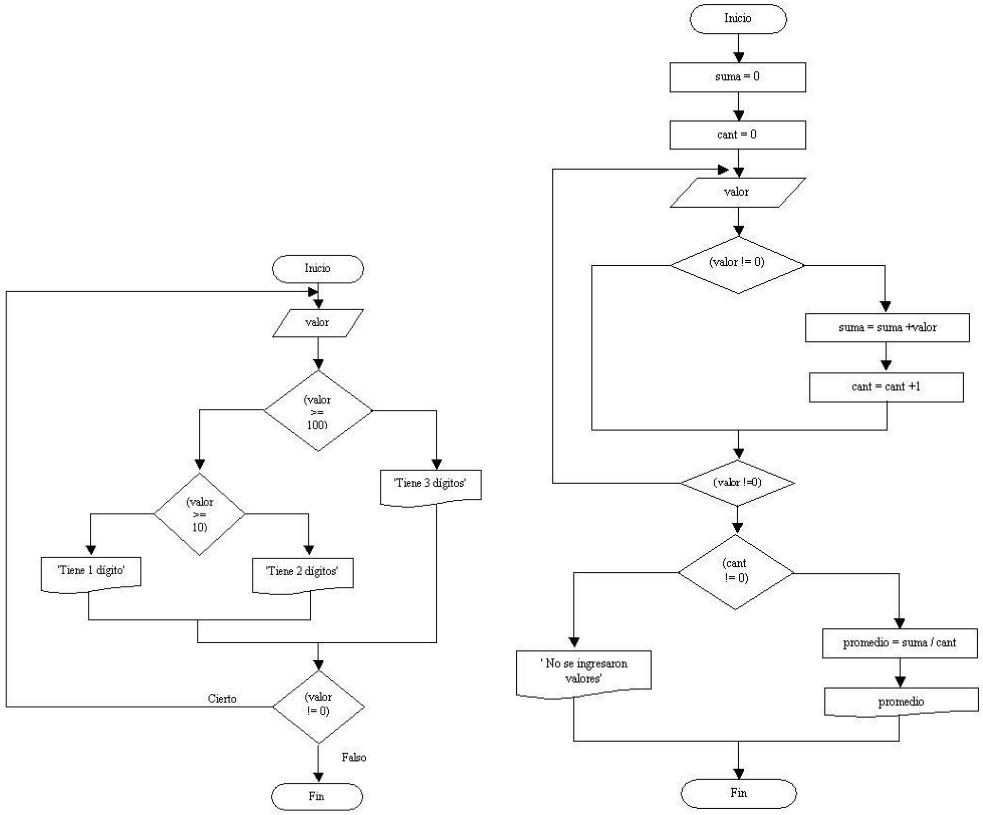
Ejemplo 1

```
Public Class Form1
   ' utilizar un cuadro de texto y un boton el codigo va en el boton el calcula cuantas
veces se necesita sumar el numero ingresado
   ' para llegar a 10 y lo imprime en un cuadro de dialogo
   Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load


   End Sub

   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Dim sum As Integer = 0
      Dim counter As Integer = 0
      Do While sum < 100
         sum = sum + CInt(TextBox1.Text)
         counter = counter + 1
      Loop
      MsgBox(CStr(counter) & " veces!" & " se sumo el mismo numero para llegar a
100 ")

   End Sub
End Class
```

Ejemplo 2

```
Public Class Form1
    Private Function Factorial(ByVal iNum As Integer) As Double
        Dim i As Integer
        i = 1
        Factorial = 1
        Do While i <= iNum
            Factorial = Factorial * i
            i = i + 1
        Loop
    End Function

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        'convertir el valor de textbox1 a valor para enviar a la funcion factorial y
nos retorne el valor
        Dim n, m As Double
        m = Val(TextBox1.Text)
        n = Factorial(Val(m))
        MsgBox(n)

    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs)

    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        MsgBox("ejercicio w while ")
    End Sub
End Class
```

Ejemplo con diagrama



5. ESTRUCTURA BUCLE FOR SIMPLE Y ANIDADA

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles y se denomina interacción al hecho de repetir la ejecución de

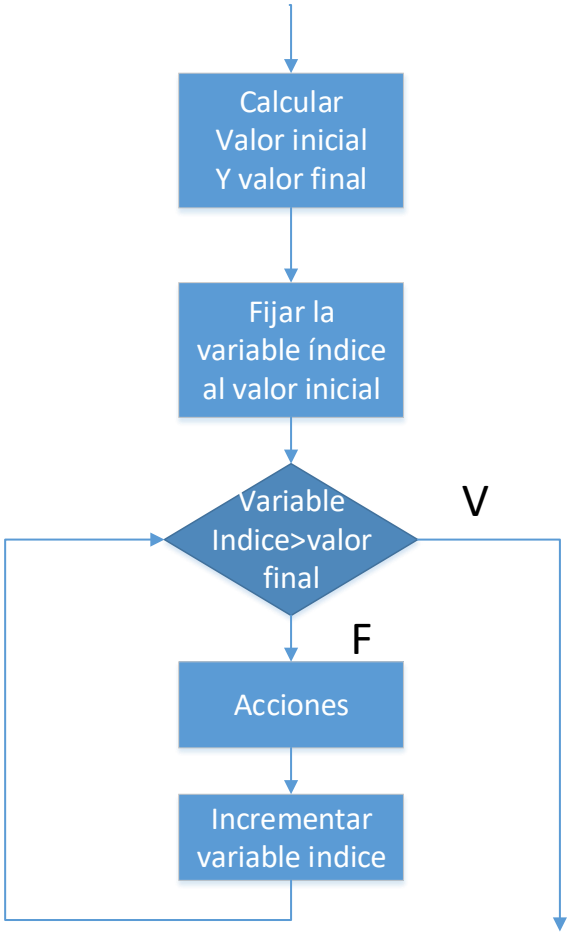
una secuencia de acciones. Las dos principales preguntas por realizarse en el diseño de un bucle son: ¿qué contiene el bucle? y ¿cuántas veces se debe repetir?

Las estructuras de bucles de Visual Basic permiten ejecutar una o varias líneas de código de forma repetitiva. Puede repetir las instrucciones de una estructura de bucles hasta que una condición sea True, una condición sea False, un número de veces especificado o una vez para cada objeto de una colección.

Bucles For…Next

La construcción For...Next ejecuta el bucle un número fijo de veces. Utiliza una variable de control de bucle, también denominada contador para realizar el seguimiento de las repeticiones. Especifica los valores de inicio y fin de este contador, y puede especificar opcionalmente la cantidad en la que se incrementa de una repetición a la siguiente.

Representación gráfica



Sintaxis

```
For counter [ As datatype ] = start To end [ Step step ]
statements
Continue For
   statements
   Exit For
   statements

Next [counter]
```

Elementos

- ✓ **Counter**
  Se requiere en la instrucción For. Variable numérica. Variable de control para el bucle.
- ✓ **Datatype**
  Obligatorio, a menos que counter ya se haya declarado. Tipo de datos de counter.
- ✓ **Start**
  Obligatorio. Expresión numérica. Valor inicial de counter.
- ✓ **End**
  Obligatorio. Expresión numérica. Valor final de counter.
- ✓ **Step**
  Opcional. Expresión numérica. Cantidad en la que se incrementa counter cada vez que se recorre el bucle.
- ✓ **Statements**
  Opcional. Una o más instrucciones entre For y Next que se ejecutan un número especificado de veces.
- ✓ **Continue For**
  Opcional. Transfiere el control a la siguiente iteración del bucle.
- ✓ **Exit For**
  Opcional. Transfiere el control fuera del bucle For.
- ✓ **Next**
  Obligatorio. Termina la definición del bucle For.

Cuando comienza un bucle For...Next, Visual Basic evalúa startend y step. Esta es la única vez que evalúa estos valores. Después asigna start a counter. Antes de ejecutar el bloque de instrucciones, compara counter con end. Si counter ya es mayor que el valor de end (o menor, si stepes negativo), el bucle For termina y el control se pasa a la instrucción que sigue a la instrucción Next. De lo contrario se ejecuta el bloque de instrucciones.

Cada vez que Visual Basic encuentra la instrucción Next, incrementa counter en el valor indicado en step y vuelve a la instrucción For. Compara de nuevo counter con end y, otra vez, ejecuta el bloque o sale del bucle, según el resultado. Este proceso continúa hasta que counter sobrepasa end o se encuentra una instrucción Exit For.

El bucle no se detiene hasta que counter ha sobrepasado end. Si counter es igual a end, el bucle continúa. La comparación que determina si se ejecuta el bloque es counter <= end si step es positivo y counter >= end si step es negativo.

Cambiar el valor de counter mientras se está dentro de un bucle puede dificultar la lectura y la depuración del código. Cambiar el valor de start, end o step no afecta a los valores de iteración especificados cuando se entró en el bucle por primera vez.

Ejemplo:

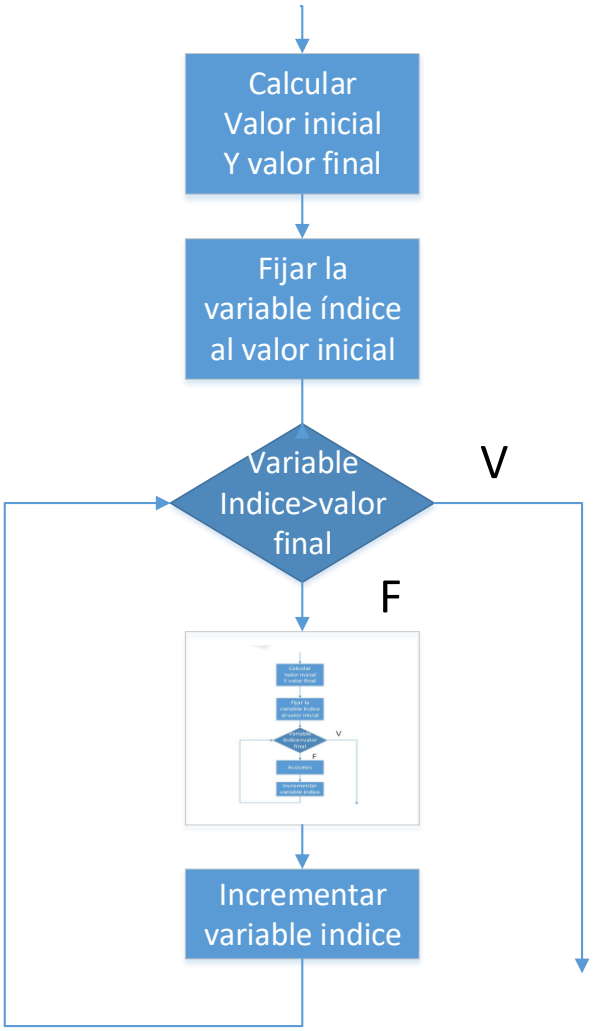
En el siguiente ejemplo se muestra el uso de la instrucción For…Next. Una variable de contador de bucle se incrementa con cada iteración del bucle. No se especifica el argumento step, así que adopta el valor predeterminado de 1.

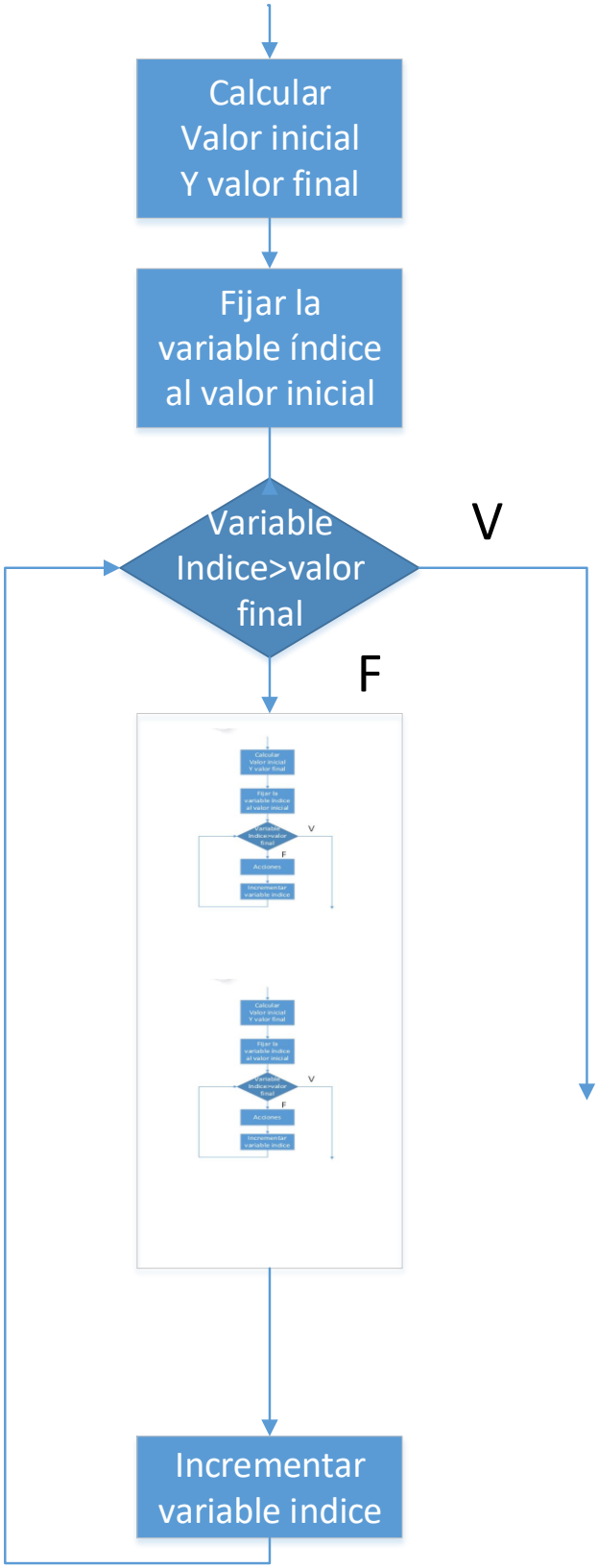
```
For index As Integer = 1 To 5
   Debug.Write(index.ToString & " ")
Next
Debug.WriteLine("")
' Output: 1 2 3 4 5
```

For anidados

Se pueden anidar bucles For colocando un bucle dentro de otro. Sin embargo, cada bucle debe tener una variable counter única.
La estructura interna de cada bucle debe estar totalmente dentro de la externa y no puede existir solapamiento.
Las variables índices o de control de los bucles toman valores de modo talque para cada valor de la variable índice del ciclo externo se debe ejecutar totalmente el bucle interno.

Representación gráfica


Calcular
Valor inicial
Y valor final
Fijar la
variable índice
al valor inicial
Variable
Indice>valor
final
V
F
Incrementar
variable indice


Calcular
Valor inicial
Y valor final
Fijar la
variable índice
al valor inicial
Variable
Indice>valor
final
V
F
Incrementar
variable indice

Ejemplo 1:

En el siguiente ejemplo, el procedimiento sumRows suma los elementos positivos de cada fila de la matriz.

```
Public Sub sumRows(ByVal a(,) As Double, ByRef r() As Double)
   Dim i, j As Integer
   For i = 0 To UBound(a, 1)
      r(i) = 0
      For j = 0 To UBound(a, 2)
         If a(i, j) > 0 Then
            r(i) = r(i) + a(i, j)
         End If
      Next j
   Next i
End Sub
```

En el ejemplo anterior, la primera instrucción Next cierra el bucle For interno y la última instrucción Next cierra el bucle For externo.

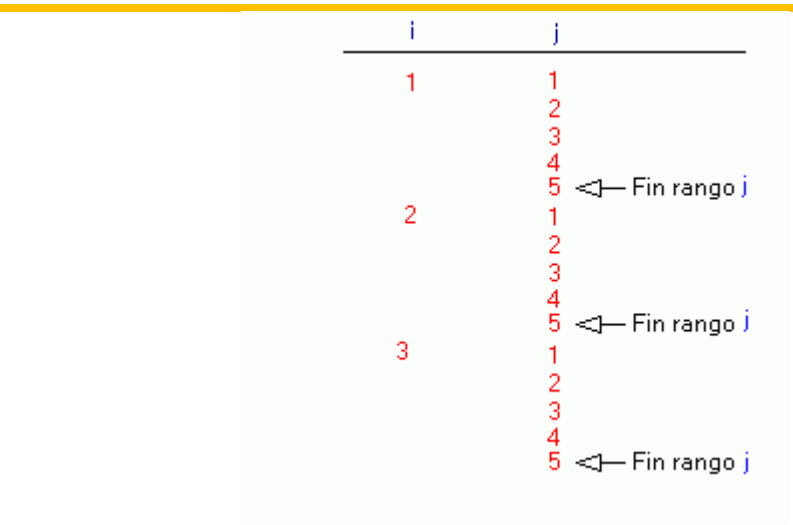
Ejemplo 2:
Supongamos que tenemos una matriz A de orden (3x5)

```
A = zeros(3,5);

for i = 1:3
   for j = 1:5
      A(i,j) = i-j;
   end;
end;
```

De esta manera asignaríamos a la variable A la matriz:

```
0  -1  -2  -3  -4
1   0  -1  -2  -3
2   1   0  -1  -2
```

El orden en que discurren los valores de "i" y los de "j" es el siguiente:



Es decir, por cada paso que da el bucle externo (el "i" en este caso) se completa el bucle interno (el "j" en este caso).

6. BUCLES FOR EACH

For Each es un bucle, el cual se relaciona a una colección o lista, este a diferencia de un For, ya tiene un inicio y un fin, y un contador de uno en uno, realiza un recorrido, por toda la lista. Donde el ciclo está determinado por el tamaño de la colección o lista (Net, 2017)

Sintaxis

For Each elemento in colección/matriz

[sentencias]

[Exit For]

[sentencias]

Next elemento

Ejemplo 1:

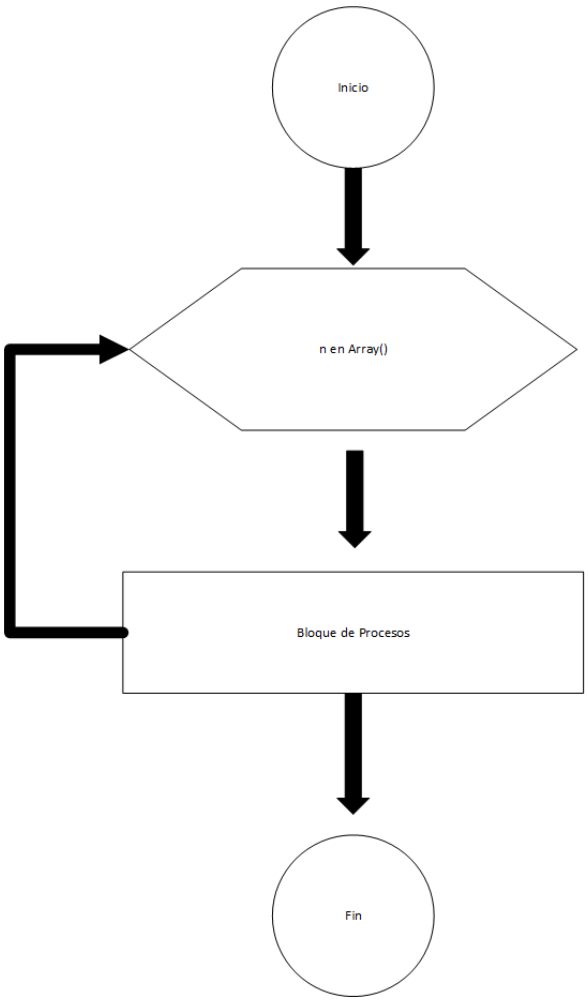
Un pequeño ejemplo, de cómo se escribe en VisualBasic

```
Dim Lista() As Integer = {1, 2, 3, 4, 5, 6, 7, 8}

    For Each i As Integer In Lista
       Button1.Text = i & Button1.Text
    Next
```

En el anterior ejemplo realiza un recorrido por todos los elementos de la colección, realiza un bucle de 8 repeticiones, donde su primer número es el arreglo 0, hasta el arreglo 7.

i viene siendo el **Array(i),** como **Array(1), Array(2),** …,

El For simple se puede también utilizar para el recorrido de una lista, pero debemos especificar el número de repeticiones, y conociendo con anterioridad el tamaño de la colección



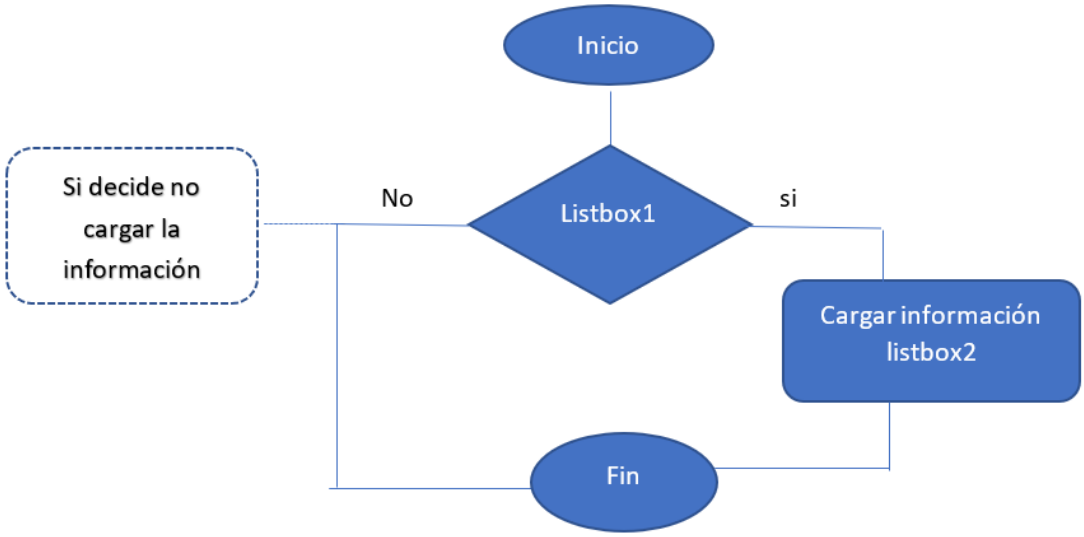
Ejemplo 2:

Cargar información de un listbox a otro con for each.

Código:

```
Private Sub cargar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cargar.Click
    For Each i As String In ListBox1.Items
        MsgBox("esto es " & i)
        ListBox2.Items.Add(i)
    Next
End Sub
```

Diagrama de flujo:



CONCLUSIONES

- ✓ Se pudo esclarecer la utilidad que brinda la sentencian For Each, y encontrar su uso o implementación

- ✓ Para la sentencia Select Case, se identificó el funcionamiento que es un bloque de múltiples condiciones

- ✓ Se pudo identificar las diferentes Estructuras de bucles que ahí en visual Basic y cuáles son sus propiedades así poder identificar e implementar la que este más acorde a analizar un ejercicio.

- ✓ Identificar el desarrollo de un ejercicio primero por su diagrama de flujo para desarrollar mejor los ejercicios teniendo claro cómo es su funcionamiento y cómo interactúan sus diversas funciones permitiendo tener un mejor código.

REFERENCIAS BIBLIOGRÁFICAS

Microsoft. (14 de 09 de 2017). Microsoft . Obtenido de Developer Network: https://msdn.microsoft.com/es-es/library/8xs8549b(v=vs.110).aspx

Net. (12 de 10 de 2017). For Each...Next Statement (Visual Basic). Obtenido de .Net: https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/for-each-next-statement

Fernández, C. (2009). Visual Basic: básico. Madrid, ES: RA-MA Editorial. Capítulo 5 Sentencias de control. Recuperado de http://bibliotecavirtual.unad.edu.co:2077/lib/unadsp/reader.action?ppg=52&docID=11046605&tm=1480460037723

OVA Unidad 2 - Fundamentos de programación

En el presente Objeto Virtual de Información, Muestra y explica el entorno de programación con sus estructuras básicas. Contiene información importante para el desarrollo de la fase 2.

Rubiano. J. (2016). Conocimiento entorno teórico [OVI]. Recuperado de http://hdl.handle.net/10596/9380

https://msdn.microsoft.com/es-es/library/2h66e7a8(v=vs.90).aspx

http://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=85&punto=&inicio=

http://www.monografias.com/trabajos70/introduccion-diagrama-flujo/image022.jpg

http://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=83&punto=&inicio=

https://www.tutorialesprogramacionya.com/visualbasicya/detalleconcepto.php?punto=10&codigo=10&inicio=0

Harvey M. Deitel, Paul J. Deitel, Cheryl H. Yaeger, Tem R. Nieto. Aug 14, 2002 by Prentice Hall. Part of the Deitel Developer Series series. Visual Basic .NET For Experienced Programmers

MSDN. Microsoft- USA. 2017 tomado de https://msdn.microsoft.com/en-us/library/aa232606(v=vs.60).aspx

MSDN. Microsoft- USA. 2017 tomado de https://msdn.microsoft.com/es-es/library/752y8abs(v=vs.120).aspx

MSDN. Microsoft- USA. 2017 tomado de https://msdn.microsoft.com/es-es/library/8y82wx12(v=vs.100).aspx5.

6. MSDN. Microsoft- USA. 2017 tomado de https://msdn.microsoft.com/es-es/library/8y82wx12(v=vs.110).aspx